

1           **Method and Apparatus for Distributed Application**  
2                           **Acceleration**

3           **FIELD OF THE INVENTION**

4           The present application relates to the field of computer  
5           networks and distributed applications. It is more  
6           particularly directed to applications that are operational  
7           on the Internet using a system of Web-browsers and  
8           Web-servers.

9           **BACKGROUND OF THE INVENTION**

10          Currently, access over the Internet to Web-based  
11          applications is provided by having a Web-browser connect  
12          directly over a network of routers to a Web-server that  
13          maintains static content in data files, and composes dynamic  
14          content by executing programs, typically cgi-bin scripts or  
15          Java servlets. However, during periods of congestion due to  
16          traffic patterns on the Internet, this arrangement results  
17          in poor response times for the end client. The situation is  
18          typically worse the farther the client is located from the  
19          Web-server and the greater the number of intermediary  
20          routers involved in the network connection.

21          One way to improve application response time, reliability,  
22          and availability is to distribute the applications to proxy  
23          servers located closer to the client browsers. Distribution  
24          of content is used to improve the performance of the network  
25          by means of proxies within the network that cache pages.

00702927-103100

1 The simple caching approach works well for data that is  
2 static and unchanging, e.g. images, video clips, etc. A  
3 proxy server is deployed within the network in many  
4 different ways. Some of the common ways include using a  
5 proxy server as a reverse proxy, where the proxy server is  
6 located closer to the web-server it is proxying for; as a  
7 forward proxy, where the proxy server is located closer to  
8 the browser or the client applications; or is as other  
9 auxiliary servers which may be located elsewhere within the  
10 network. The proxy server usually provides information to  
11 the browser on behalf of a backend server. The browser may  
12 contact the proxy server due to a variety of reasons e.g.,  
13 because it has been explicitly configured to do so, or  
14 because the domain name server gives it the location of a  
15 proxy server instead of the backend web-server, or because a  
16 network operator or backend web-server operator has  
17 configured the network to send requests from the browser to  
18 the proxy server in a transparent fashion.

19 However, the techniques of caching that are commonly  
20 deployed in the current Internet do not work well with a  
21 large portion of web-accessible content. Data that is  
22 personalized to a client, or data that is generated by  
23 invocation of programs like cgi-bin scripts or servlets can  
24 not be readily cached at the proxies. For a server offering  
25 electronic services over the Internet, non-static data forms  
26 a significant portion of their overall data content. It  
27 would be advantageous to have a scheme whereby such  
28 dynamically generated content, and web-centric applications  
29 can also benefit from the presence of proxies.

1 As in the case of caching of static data, it is highly  
2 desirable that the caching of applications be done so that  
3 the administrative and operation control of the  
4 data/application resides with the original server, rather  
5 than with at the proxy server. A solution is needed which  
6 accelerates applications while still providing the  
7 administrative control of the application to the original  
8 server, rather than the proxy server.

9 SUMMARY OF THE INVENTION

10 Accordingly, an aspect of the present invention presents  
11 methods and apparatus by which to accelerate execution of  
12 Web front-ended applications by means of executing them at  
13 proxies located closer to the client browsers.

14 Another aspect of the present invention presents methods and  
15 apparatus for a proxy server which provides an execution  
16 environment for acceleration of Web front-ended  
17 applications.

18 Another aspect of the present invention presents methods and  
19 apparatus for a backend server which provides an execution  
20 environment for acceleration of Web front-ended  
21 applications.

22 BRIEF DESCRIPTION OF THE DRAWINGS

23 These and other aspects, objects, features, and advantages  
24 of the present invention will become apparent upon further

09702927.103100  
DOCKET

1 consideration of the following detailed description of the  
2 invention when read in conjunction with the drawing figures,  
3 in which:

4 Fig. 1 is a block diagram showing an example of an  
5 application acceleration infrastructure, and their manner of  
6 interaction in accordance with the present invention;

7 Fig. 2 is a block diagram of an example showing steps for a  
8 method of application acceleration in accordance with the  
9 present invention;

10 Fig. 3 is a block diagram of an example showing steps taken  
11 by a proxy server in executing a requested service for an  
12 end client in accordance with the present invention;

13 Fig. 4 is a block diagram of an example showing steps for  
14 determination of an appropriate set of programs to run in  
15 response to a particular end client service request in  
16 accordance with the present invention;

17 Fig. 5 is a block diagram of an example showing structure  
18 of an information record, used in a solution to the  
19 distributed application acceleration problem in accordance  
20 with the present invention;

21 Fig. 6 is a block diagram showing an example of a structure  
22 of a distributed system to implement a solution to the  
23 distributed application acceleration problem in accordance  
24 with the present invention;

1 Fig. 7 is a block diagram showing an example of a structure  
2 of a proxy server, used in a solution to the distributed  
3 application acceleration problem in accordance with the  
4 present invention; and,

5 Fig. 8 is a block diagram showing an example of a structure  
6 of a backend server, used in a solution to the distributed  
7 application acceleration problem in accordance with the  
8 present invention.

9 DETAILED DESCRIPTION OF THE INVENTION

10 Figure 1 shows an example of the components of an  
11 application acceleration infrastructure. A communication  
12 network 101 is used to interconnect client devices  
13 containing Web-browsers like 102 and 103 to a backend web  
14 server 104 that provides Web-based applications. Major  
15 portions of the application code are moved onto proxy  
16 servers like 105 and 106 that are in closer proximity to the  
17 respective end client devices. Communications is maintained  
18 between the proxy servers 105 and 106 and the backend server  
19 104 so that the backend server can continue to exercise  
20 administrative control over the distributed portions of the  
21 applications code. Communications is also maintained  
22 between the end client devices 102 and 103 and the backend  
23 server 104 so that the latter can continue to provide  
24 applications services that are not readily distributable.

25 Fig. 2 delineates the steps that comprise a method of  
26 application acceleration. In response to a client service  
27 request 201, a Wide Area Load Balancing module determines

09702927-103100

1 the appropriate proxy server to which to direct the request  
2 based upon current network performance characteristics and  
3 the location of the end client client device as in 202. The  
4 proxy server then determines the set of programs that it  
5 needs to run in order to satisfy the end client request as  
6 in 203. These may be already resident at the proxy server  
7 if they had been needed to satisfy previous requests from  
8 other users, or if they had been deployed to the proxy  
9 server in anticipation of end client requests for basic or  
10 popular applications. In any of these cases, they are  
11 resident at the backend server. The proxy server obtains  
12 the data and application code appropriate for fulfilling the  
13 service request as in 204, and executes the indicated  
14 functions for the end client on behalf of the backend server  
15 as in 205.

16 Fig. 3 is a block diagram that shows an example of steps  
17 taken by a proxy server in executing a requested service for  
18 an end client. The response from an application is often  
19 tailored to a particular end client or class of end users,  
20 depending upon organizational affiliations or other  
21 criteria. The proxy server thus obtains information from  
22 the backend server regarding which execution parameters to  
23 employ in satisfying this specific end client request as in  
24 301. The application logic is then executed as in 302, and  
25 any administrative information or error messages resulting  
26 from the application execution are logged and sent to the  
27 backend server as in 303. In order to simplify systems  
28 management and provide support for problem tracking and  
29 diagnosis, the backend server maintains responsibility for  
30 the application logic. This allows the proxies to be shared  
31 effectively by many back end servers without increasing

1 their complexity. The results of the execution of the  
2 application logic are then sent to the end client as in 304.

3 Fig. 4 is a block diagram of steps for a determination of  
4 an appropriate set of programs to run in response to a  
5 particular end client service request. The request  
6 indicates a target location within the backend server for  
7 the application logic, typically in the form of a Uniform  
8 Resource Locator (URL). The proxy server maintains data on  
9 application programs that it represents, and determines what  
10 service is being requested as in 401. The appropriate  
11 proxylet record is then retrieved as in 402. The proxylet  
12 record contains a data field specifying the set of programs  
13 needed in order to properly satisfy an end client service  
14 request, and this is read by the proxy server as in 403.

15 Figure 5 shows the structure of a proxylet record used in  
16 order to determine the set of programs that are to be  
17 executed at a proxy when a request is received from a  
18 client. The proxylet record 501 includes several fields.  
19 The RequestedURL field 503 contains the URL which the client  
20 was requesting. The ExecuteURL field 505 contains the name  
21 of a URL which will be executed at the local proxy in order  
22 to support this request. The location where the compute  
23 code required to run ExecuteURL is contained in the field  
24 CodeLocation 507. The ParameterList field 509 contains any  
25 parameters which would be passed to the program in order for  
26 it to be executed. The contents of the ParameterList field  
27 509 may be same for different proxies within the network, or  
28 they may be different for multiple proxy servers. The  
29 ExpirationTime field 511 contains the date until which this  
30 proxylet record may be considered valid. Once the time

DOCKET NUMBER: YOR92000757US1

1 specified by field 511 has expired, a proxy server would  
2 need to retrieve a new proxylet-record from the backend  
3 server. The LoggerURL field 513 identifies a location where  
4 the error messages and diagnostic output of the proxylet are  
5 provided. Typically, the loggerURL would identify a  
6 location on the backend server. The codeVersion field 515  
7 contains the time when the program set identified in the  
8 field codeLocation 505 was last modified. Other fields may  
9 also be included within the proxylet-record as used in  
10 different embodiments of the invention.

11 As an example, let us consider a request from a client which  
12 is targeted to the location

13 http://main-server.com/servlet/program1.

14 This request is delivered to the proxy server running at the  
15 machine proxy-server.com. The proxylet record for this  
16 request might contain the fields of requestedURL field being

17 http://main-server.com/servlet/program1,

18 with executeURL field being /servlet/proxy-program1, the  
19 codeLocation field being

20 http://main-server.com/proxylet/prox-program1,

21 the ParameterList being empty, and the LoggerURL being

22 http://main-server.com/servlet/logger,



1 the expirationTime field being 30000 seconds after a  
2 reference date such as 1/1/1970, and the codeVersion field  
3 being 29500 seconds after a reference date such as 1/1/1970.

4 When such a request is received at the machine  
5 proxy-server.com, the machine checks if it has a cached  
6 entry corresponding to the proxylet-record where the  
7 requestedURL matches the request being received. If it  
8 does, it checks the expirationTime field to ensure that the  
9 record needs to be updated. It then checks to see if it has  
10 the program identified by the executeURL installed locally  
11 at the proxy-server. If it does, then it runs the program  
12 passing to it any parameters contained in the parameterList  
13 field. If the proxylet-record is not found, or if the  
14 current time is greater than the time specified in the  
15 expirationTime field, the proxy-server contacts the  
16 main-server to obtain a fresh copy of the proxylet-record  
17 prior to executing the steps outlined above.

18 The schemes as described above can be seen as a distributed  
19 system that achieves acceleration of applications by  
20 distributing their execution. An apparatus 601 that  
21 implements the distributed system is shown in Figure 6.  
22 Figure 6 includes three distributed components, a wide-area  
23 load balancer 603, a application distributor 605, and a base  
24 class library 607. The wide area load-balancer 605 is a  
25 distributed module which collects performance statistics  
26 about the network, and determines the most appropriate place  
27 where a request should be sent out to. Since these  
28 components are distributed components, the connectivity  
29 information among them is not shown in the Figure. However,

1 the connectivity information will become clearer from the  
2 description provided below.

3 The Wide Area Load Balancer 603 is a component responsible  
4 for distributing client requests to different proxy servers  
5 within the network. It can be implemented in a variety of  
6 manners. One common way to implement it is by means of a  
7 modified domain name server. The domain name server,  
8 usually abbreviated to DNS server, is the application in the  
9 network responsible for mapping machine names to IP  
10 addresses. A modified domain name server can return an IP  
11 address which corresponds to an appropriate proxy server  
12 when a client requests an address for the backend server.  
13 The appropriate proxy server is determined on the basis of  
14 the current network performance characteristics and the  
15 location of the client.

16 An alternative implementation of the wide area load balancer  
17 includes a module within the backend server that is  
18 responsible for redirecting requests to the appropriate  
19 proxy server. Such a redirection module might be  
20 implemented as a plug-in module among a variety of  
21 web-servers such as Apache, NetScape or Microsoft IIS  
22 server, which are commonly in use in the industry. The  
23 module would look at a table of redirection rules, which  
24 specify how requests coming from specific client IP  
25 addresses should be dispatched, and use this information to  
26 determine the appropriate proxy-server to which the request  
27 should be dispatched. The selection of the proxy-server can  
28 be based on other criteria included in the rule, e.g. The  
29 resource (URL) being requested by the client, or a cookie  
30 which is contained within the client's request.

001100 " 103100 265222

1 Another embodiment of the wide area load balancer uses a  
2 stand-alone http server which provides the same  
3 functionality as that of the module described above. The  
4 http server implements the ability to direct requests to  
5 proxy-servers, or to another server operation locally at the  
6 site with the stand-alone http server, which services  
7 requests that need to be performed locally.

8 The application distributor 605 is responsible for ensuring  
9 that the set of programs that need to be executed at the  
10 proxy server are indeed available at that server. There are  
11 many embodiments of such an application distributor which  
12 will be useful to those skilled in the art.

13 One useful embodiment of an application distributor uses a  
14 program which keeps track of all the programs and data that  
15 is available at a main-server, and maintains a replica of  
16 those program and data at the proxy server. Such an  
17 distributor will push the changes that occur at the  
18 main-server out to the proxy servers in order to maintain  
19 this consistency of program and data.

20 Another embodiment of an application distributor uses a  
21 program that runs at the proxy-server and caches a copy of  
22 programs and data from the main-server when requests that  
23 would cause execution of those programs are received at the  
24 proxy-server. The data that the programs need to execute is  
25 also retrieved as needed, and cached at the proxy-server.  
26 The programs and data are both cached on demand.

DOCKET NUMBER: YOR920000757US1

1 Yet another embodiment of an application distributor uses a  
2 program that employs both of these techniques. Some of the  
3 programs which are most often used are pushed out to all of  
4 the proxy-servers while other programs are cached on demand  
5 at the proxy-server.

6 The class library 607 is a set of programs that exists at  
7 all the proxy-servers and the backend servers. It contains  
8 a collection of classes that enable many functions to occur.

9 One of the classes contained in the library identifies the  
10 set of programs that are capable of executing at the  
11 proxy-server. All such programs are derived from a specific  
12 class proxylet, and the fact that these programs are  
13 subclassed from the class proxylet is used to validate that  
14 the program can execute at the proxy-server. Yet another  
15 class provided in the class library is the Logger class,  
16 which allows the output and error messages generated by the  
17 program executing at the proxy-server to be copied to the  
18 main-server for purposes of logging and diagnostics. Yet  
19 other set of classes allow for the caching of different  
20 types of application data. Instances of these include the  
21 programs for caching queries made to a directory or a  
22 database, programs for caching records in a database, as  
23 well as programs for caching files.

24 The components of the distributed architecture shown in  
25 Figure 6 are various proxy-servers and the backend server.  
26 A proxy-server which is such a component in this solution is  
27 shown in Figure 7. The proxy-server 701 includes a Cache  
28 Manager 703, a set of cached information records 705, a set  
29 of cached programs 707, and a set of cached data 709. The  
30 Cache Manager 703 is responsible for managing and updating

DOCKET# 2520260

1 the different types of caches, namely the set of cached  
2 information records 705, the set of cached programs 707 and  
3 the set of cached data 709. The cache manager maintains all  
4 of these caches in an appropriate manner. The set of cached  
5 information records 705 contains information about the  
6 cached programs that are cached locally in the set of cached  
7 programs 707. The format of information contained in the  
8 information record is similar to that of the proxylet record  
9 501 shown in Figure 5. The Cache Manager 703 utilizes the  
10 services of an application distribution module 605 in order  
11 to ensure the consistency and coherence of the different  
12 sets of caches. The set of cached data programs 709 is  
13 maintained current by using data caching techniques that are  
14 well-known to those versed in the art.

15 Figure 8 shows the structure of a backend server which would  
16 respond to the proxy server shown in Figure 7 and provides  
17 the other part of the infrastructure for application  
18 acceleration. The backend server 801 includes a traditional  
19 web-server 803, a set of programs to be downloaded to proxy  
20 servers 805, a set of local programs 807, and a set of  
21 operational programs 809. The web-server 803 provides the  
22 means by which a proxy server can gain access to the set of  
23 programs 803, 805 and 807. The set of downloadable programs  
24 803 are transferred to a proxy server if it makes a request  
25 for them. All or a subset of the programs may be  
26 transferred to the proxy server. The set of local programs  
27 807 provides a means by which a proxy server can execute  
28 some parts of the processing at the backend server itself.  
29 As an example, a proxy server may want to execute programs  
30 related to updating databases only at the backend server.  
31 The set of operational programs 809 provides a means by

007207.103100

1 which a proxy-server can provide diagnostics and management  
2 information to the backend server. An example of an  
3 operational program 809 would be a logger servlet that can  
4 obtain logging messages generated by programs executing at  
5 the proxy server.

6 In some embodiments of the backend server, the web-server  
7 may incorporate an ability to redirect client requests to  
8 other servers. This would be an instance of the application  
9 distribution module. In other embodiments, the backend  
10 server may rely upon the domain name service to do such  
11 redirections. The backend server as described in Figure 8  
12 and a set of proxy servers as described in Figure 7 together  
13 provide the infrastructure for distributed application  
14 acceleration.

15 It is noted that the present invention can be realized in  
16 hardware, software, or a combination of hardware and  
17 software. A tool according to the present invention can be  
18 realized in a centralized fashion in one computer system, or  
19 in a distributed fashion where different elements are spread  
20 across several interconnected computer systems. Any kind of  
21 computer system - or other apparatus adapted for carrying  
22 out the methods described herein - is suitable. A typical  
23 combination of hardware and software uses a general purpose  
24 computer system with a computer program that, when being  
25 loaded and executed, controls the computer system such that  
26 it carries out the methods described herein. The present  
27 invention can also be embedded in a computer program  
28 product, which comprises all the features enabling the  
29 implementation of the methods described herein, and which -

1 when loaded in a computer system - is able to carry out  
2 these methods.

3 Computer program means or computer program in the present  
4 context includes any expression, in any language, code or  
5 notation, of a set of instructions intended to cause a  
6 system having an information processing capability to  
7 perform a particular function either directly or after  
8 either conversion to another language, code or notation;  
9 and/or reproduction in a different material form.

10 It is noted that the foregoing has outlined some of the more  
11 pertinent objects and embodiments of the present invention.  
12 This invention may be used for many applications. Thus,  
13 although the description is made for particular arrangements  
14 and methods, the intent and concept of the invention is  
15 suitable and applicable to other arrangements and  
16 applications. It will be clear to those skilled in the art  
17 that modifications to the disclosed embodiments can be  
18 effected without departing from the spirit and scope of the  
19 invention. The described embodiments ought to be construed  
20 to be merely illustrative of some of the more prominent  
21 features and applications of the invention. Other  
22 beneficial results can be realized by applying the disclosed  
23 invention in a different manner or modifying the invention  
24 in ways known to those familiar with the art.

0070297.103100